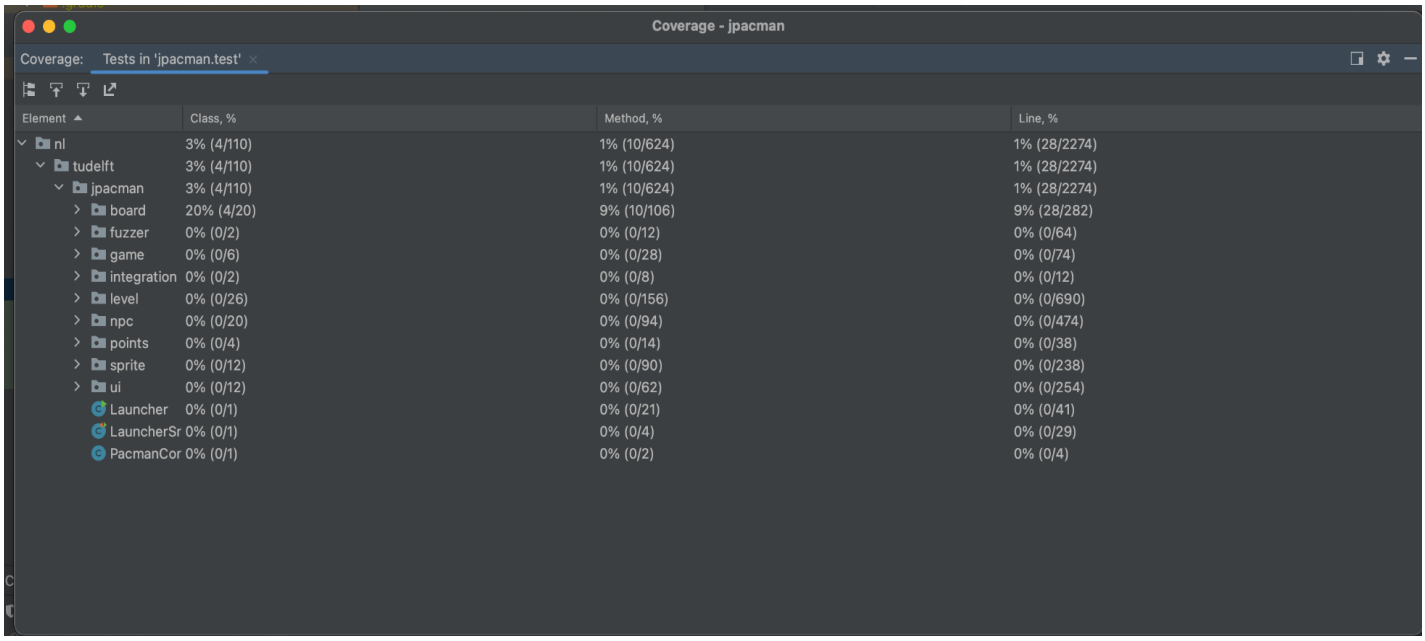
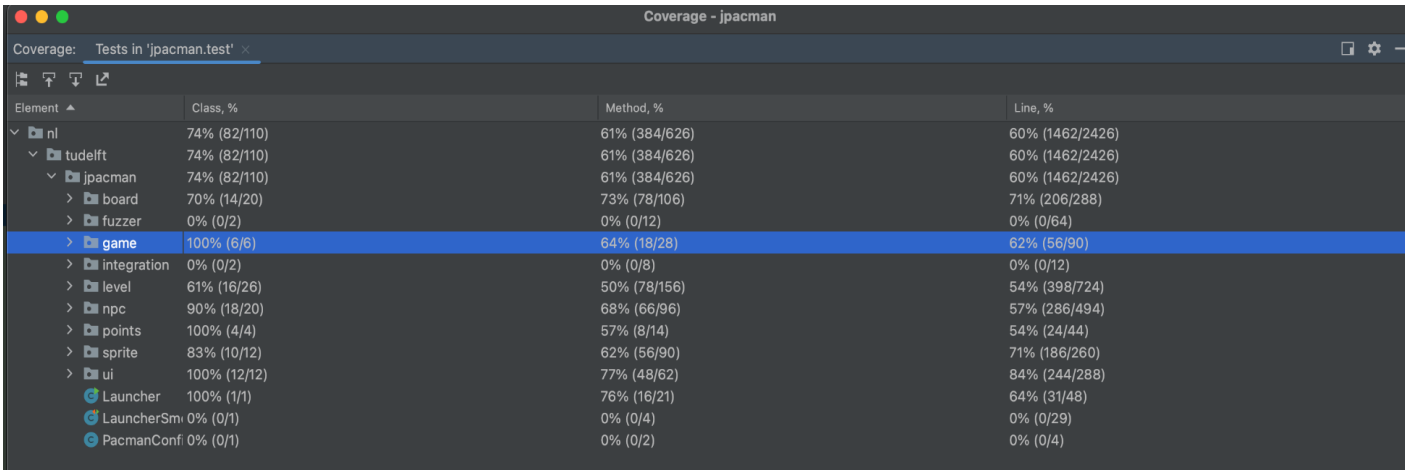
**Task 2.1:**

This is the IntelliJ coverage prior to any changes made:



This is the IntelliJ coverage after I developed a test for isInProgress() for the Game.Java file:

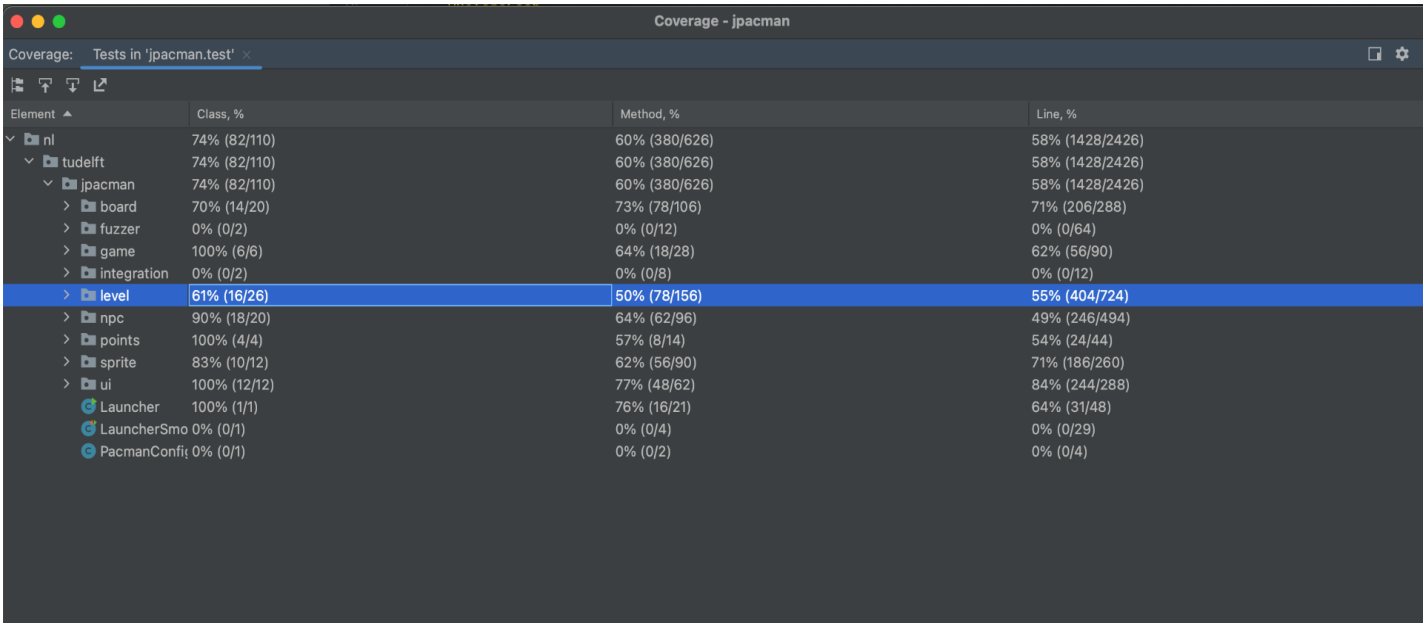



Here is the code for this test:

```
public class InProgressTest {

    private Launcher launcher;

    @Test
    void testIsInProgress() {

        launcher = new Launcher();
        launcher.launch();

        Game game = launcher.getGame();
        game.start();

        assertThat(game.isInProgress()).isEqualTo(true);
    }
}
```

This test launches the game and checks if it is running to confirm the in progress function. This test created a big difference in coverage moving up from 3% to 74% for class, from 1% to 61% for method, and from 1% to 60% for lines covered.

This is the IntelliJ coverage after I developed a test for isInProgress() for the Game.Java file:

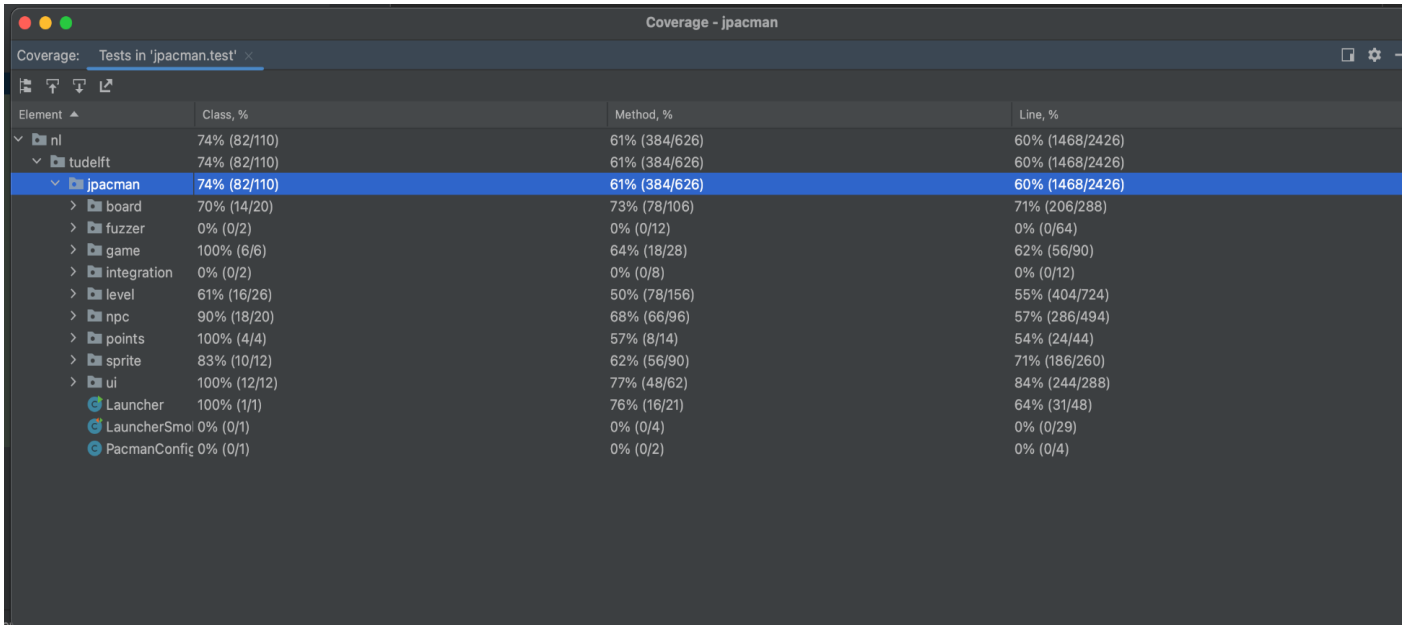



Here is the code for this test:

```
void testInProgress() {

        level = new Level(board, Lists.newArrayList(ghost),
Lists.newArrayList(
            square1, square2), collisions);
        level.start();

        assertThat(level.isInProgress()).isEqualTo(true);
}
```

This test creates a new level and then runs the test to check if the level is in progress. This test did not have as big of an impact. Level increased by 1% for lines, and that was the only improvement.

This is the IntelliJ coverage after I developed a test for getInterval() for the Ghost.Java file:




Here is the code for this test:

```
void testInProgress() {

        level = new Level(board, Lists.newArrayList(ghost),
Lists.newArrayList(
            square1, square2), collisions);
        level.start();

        assertThat(ghost.getInterval()).isNotNull();
}
```

This test first establishes a new level and then confirms that the getInterval function returns a non null value. This test added slight improvements with a 1% increase in method and a 2% increase in lines covered.

**Task 3.0:**

The coverage from JaCoCo is not similar to the stats that I received from IntelliJ. The one main similarity is that both list the coverage at 74% for the main. Other than that, every other percentage shown is different. On JaCoCo, the information is listed out more visually and I can have an easier time understanding what branches and instructions were covered.

I think that the visualization of the uncovered branches is particularly useful, because it is information that I did not easily have through IntelliJ.

I prefere INtelliJ's coverage window. I think that the reason that I like this window is because it presents the information in a more clear way. With the JaCoCo, the vizualization is helpful, but I'm not entirely sure what all of it means. I think that with the IntelliJ coverage it is easier to see how your tests are performing.